Content 18

Pointers In C

Before discussing about pointers let me tell you when we define and initialize a variable at that time we come to know about these things :

* Memory block i.e. variable get some space in RAM and we can think of that as a block.
* Name of memory block or Variable’s name
* Content of that block i.e. value in that variable
* Address of memory block i.e. unique address which allows us to access that variable.

We can print address of any variable by using printf function as :

printf(“%d”,&variable\_name);

**Pointer :**

“*Pointers are nothing just a variable which stores the address of other variables and by using pointers we can access other variables too and can even manipulate them*”.

* Pointer is a variable that contains address of another variable. It means it is a variable but this variable contains address or memory address of any other variable.
* It can be of type int, char, array, function, or any other pointer.
* Its size depends on architecture.
* Pointers in C Language can be declared using \*(asterisk symbol).

**Pointer Syntax;**

Here is how we can declare pointers.

int\* p;

Here, we have declared a pointer p of int type.

You can also declare pointers in these ways.

int \*p1;

int \* p2;

Let's take another example of declaring pointers.

int\* p1, p2;

Here, we have declared a pointer p1 and a normal variable p2

**Get Value of Thing Pointed by Pointers**

To get the value of the thing pointed by the pointers, we use the \* operator. For example:

int\* pc, c;

c = 5;

pc = &c;

printf("%d", \*pc); // Output: 5

Now let’s see about some of the operators which we use with Pointers :

* **Address of Operator (&) :**
* It is a unary operator.
* Operand must be the name of the variable.
* & operator gives address no. of variable.
* & is also known as “Referencing Operator”.

* **Indirection Operator :**
* \* is indirection operator.
* It is also known as “Dereferencing Operator”.
* It is a unary operator.
* It takes address as an argument.
* \* returns the content/container whose address is it’s argument.

#include<stdio.h>

int main()

{

int a=5;

printf("%d\n",&a );

printf("%d",a );

return 0;

}

In above example you will see that we printed variable ‘a’ address and its value. So in second printf statement you can see we used two unary operators i.e. \* and "&" operator. As we know unary operator associativity is from right to left so first of all & of operator will be resolved and then the address of variable ‘a’ will be the argument of \* operator. That’s how we can print values or can use these Address of Operator and Indirection Operator.

#include<stdio.h>

int main()

{

int x,\*a;

return 0

}

When \* (indirection operator) is written before any pointer variable like (\*a) then that whole variable resembles like original variable i.e. the variable to whom the pointer is pointing.

Or,

We can say that \*a pointer becomes the variable whose address is in a pointer.

#### ****Uses of Pointers:****

* + Dynamic Memory Allocation
  + Arrays, Functions and Structures
  + Return multiple values from a function
  + Pointer reduces the code and improves the performance

That’s all about basics of Pointers.

**Code1 for Explaining Pointers:**

#include <stdio.h>

int main()

{

    int x = 5;

    int \*ptra = &x; //&x is for storing the adress of x or pointing x.

    printf("The value of x is %d \n", x);

    printf("The value of x by pointer is %d \n", \*ptra);

    printf("The adress of x is %p \n", &x);  //printing adress of x

    printf("The adress of pointer to x is %x \n", &ptra); //%x is for printing address of pointer but Hexadecimal number

    printf("The adress of pointer to x is %p", ptra); //%p is for printing address of pointer

    return 0;

}

**Output:**

The value of x is 5

The value of x by pointer is 5

The adress of x is 0061FECC

The adress of pointer to x is 61fec8

The adress of pointer to x is 0061FECC